Quasi-optimal Coverage Algorithm for Simple Robot in an Unknown Environment

Dominique Duhaut

Quasi-optimal Coverage Algorithm for Simple Robot in an Unknown Environment

Dominique DUHAUT
Lab-STICC-CNRS
Université de Bretagne Sud, France
dominique.duhaut@univ-ubs.fr

Abstract

In this paper we present a simple algorithm to make a quasi-optimal coverage of an unknown environment. Under some hypothesis, we demonstrate that it is possible to build a map of the environment and to cover it, without passing two times in the same space, except if it's impossible and we will describe these cases and show that they are very limited.

The robot is supposed to be a non holomomic platform with 2 powered wheels for "tank-like" movements, and a set of 7 distance sensors in front of the robot.

Introduction

In robotics, the problem of the covering the space in an unknown environment is a very standard problem. On this the first relevant synthesis can be found in [1]. SLAM simultaneous localization and mapping can be coupled to cover unknown space [2]. Based on graph description [3] and using a research graph algorithm is another method used to coverage. Another work proposes the Boustrophedon cellular decomposition [4] to build a path for space coverage.

Usually in this paper the sensors are a set of sensor surrounding the robot [5] or a laser sensor [6]. In this paper we will use a very simple sensor.

In this paper we will describe the robot used in the simulation and show that in a discretized environment there is no optimal path to cover the space. We then define the quasi-optimal notion. In part two we present the quasi-optimal algorithm and results of simulation. We finally discuss some perspectives to this work.

1. The robot and the environment

Description of the robot

Like a large set of commercial robots, our work uses a "tank-like" robot with two independent wheels.

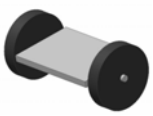

Fig.1. Non holomomic platform with 2 powered wheels for tank-like movements

In this paper we consider that the robot can only move in three directions: front, left, right in a discretized environment decomposed in elementary suares. We assume that the robot fully covers on square.

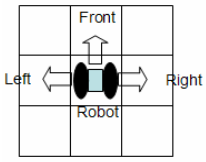

Fig.1. The 3 possible displacements

The robot is supposed to be equipped with distance sensors able to detect an obstacle in the 7 places in front of the robot and measuring a distance of 3 squares.

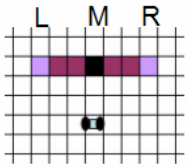

Fig.3. Line of 7 sensors for obstacle detection in front of the robot

Hypothesis on the environement

The hypotheses on the environment are:

- There is no dynamical object in the space, the robot is the only element moving
- The space is large relative to the robot. This means that if the robot covers a elementary square then the minimal distance between two static obstacles is six times this distance: six squares.

In this kind of environment it is possible to decompose the space in rectangles [2] that represents the free space.

The figures 3&4 show that this rectangle decomposition is not unique

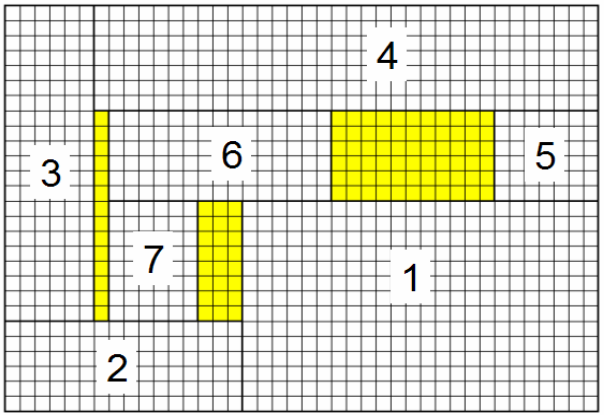

Fig.3. Decomposition of the space in 7 rectangles

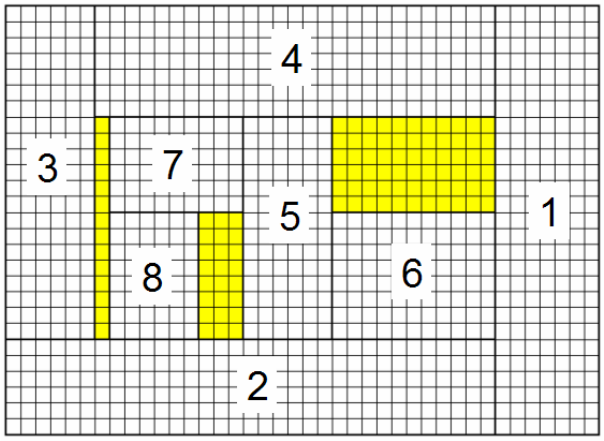

Fig.4. Decomposition of the space in 8 rectangles

Has shown in the proposed algorithm in part 2 the rectangle decomposition will depend of the initial position of the robot. In the following, we will consider that each of the 8 rectangles of the space fig 4 are empty rooms and that a door exist between two adjacent rectangles.

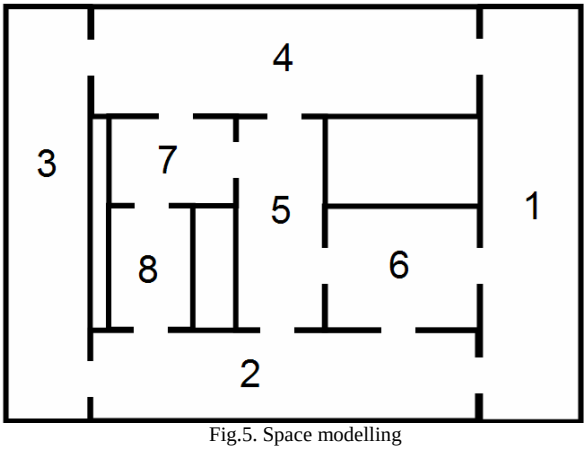

Fig.5. Space modelling

We will then consider that the environment is a set of rooms in which there are communication doors, has presented in fig 5.

No optimal path for covering the space

In this section we want to show that with the type of "tank-like" robot is is not possible to give an optimal path to cover the surface if we fix and entry door and an exit door. By optimal path we mean that the robot passes by all the squares of the environment one time and only one time.

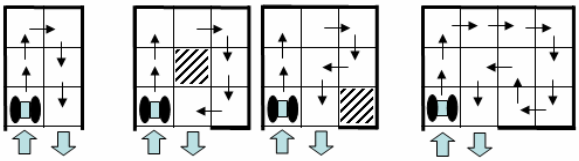
Fig.6. Depending on the size of the environment, this kind of robot cannot find an optimal path

Has shown in the figure 6, with the type if displacement allowed by the robot, an optimal path depends on the number of squares to cover and the position of the entry and the exit door. In the case of 6 or 12 squares a solution exists. But with a 9 squares there is no solution. We have to notice that the result could be different if we change the place or the entry and the exit door see figure 7.

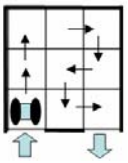
Fig.7. The entry and exit position are also parameters that influences an optimal path

We will define a quasi-optimal path as a path covering the environment, covering a maximum number of squares, without passing two times by the same square and which lets a minimal number of squares non visited.

2. The quasi-optimal algorithm

In this part, we present the principle of the algorithm, the next section will show why and when it is not optimal but only quasi-optimal. The algorithm is in two parts the first one explores the environment and build the corresponding map, the second part will finish the covering of the non visited space and will visit the adjacent rooms while passing in front of a doors.

Part 1: Building the map between obstacles

The robot enters in the room and covers the centre of the space and lets a walkway around the walls of 2 squares large.

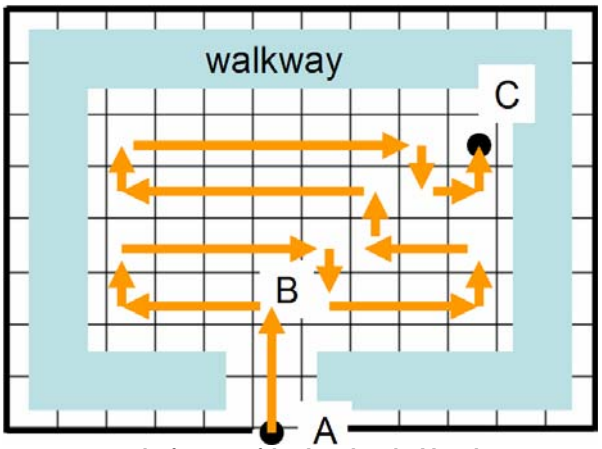

Fig.8. The first part of the algorithm : building the map

The robot enters in A in the first square then moves to the initial position B. Then a boustrophedon [4] path is generated until the sensors detect an obstacle. This path ends in the C position. Due to the distance of obstacle detection the walkway of two squares large as not been visited by the robot (except between A and B).

Part 2 Finishing the coverage

In the second part the robot uses the walkway to access to all the doors surrounding the room and ending the coverage.

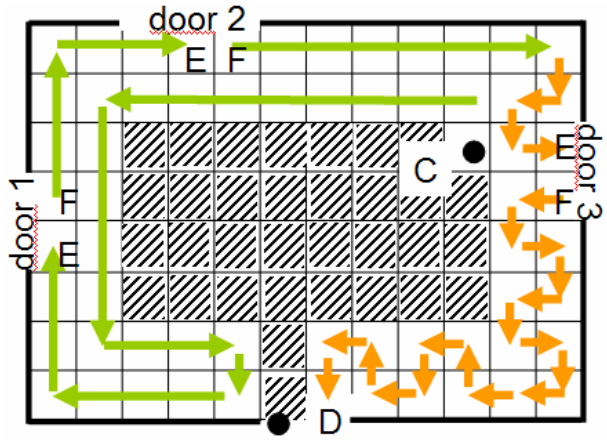

Fig.9. The second part of the algorithm : finishing the coverage

From the C position the robot moves in the opposite direction of the exit door D to finish covering the space. Following the path if a door, connecting to a non visited room, is founded then the robot enters the new room E and will re-enter coming back from this room in the next square of the space F. Then the displacement continues.

When the robot exceeds the C position then again a boustrophedon path is generated. If a door if founded then the robot enter this new room at a position E and comes back from the room at the next position F. This is done until the exit position D is reached.

3. It is only quasi-optimal

We run a simulation and show that, as mentioned in fig 11 depending on the geometry of the environment the robot will construct two different freeways: one strictly 2 squares large, the second one depending of the position of the obstacles in the environment will be partially 3 squares large.

Part 1: Building the map between obstacles

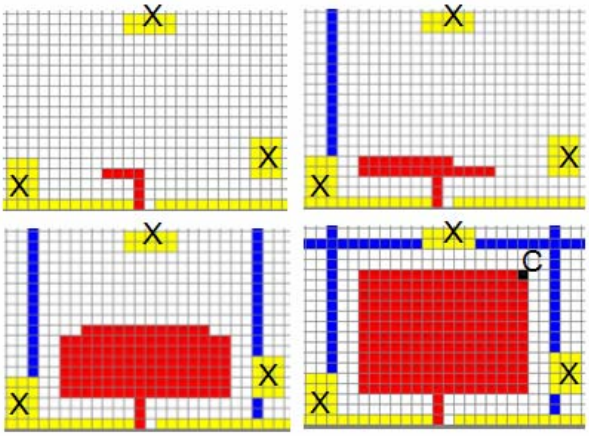

Fig.10. Screen copy of the simulation of the map building

The obstacles are in yellow in the fig 10 and when the robot finds an obstacle in front or on the side, it extends the position of the obstacle by a line (in blue) to figure the limits of the visited “room” see figure 5. If the obstacles are at good position (odd squares horizontal) then the exploration finishes with a covered rectangle (in red) and the robot is in C an angle, the freeway around this rectangle is exactly 2 squares large.

But depending of the distance between obstacles we can have 4 kinds of covering fig 11.

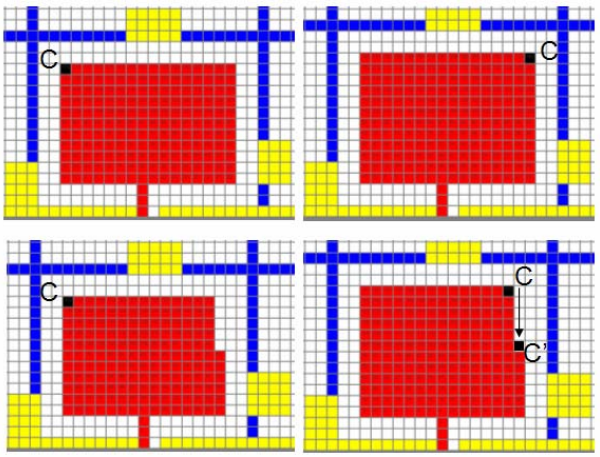

Fig.11. The visited space depending in the position of the obstacles.

In the top of the figure 11 we can see that for the same distance in columns between two obstacles (left and right) the covered surface is a rectangle, the robot finishing in C position on the left or on the right depending on the distance in line between two obstacles (top and down).
On the bottom of fig 11, if we increase the horizontal distance in column of one square then the problem mentioned in fig 6 appears and then the robot must increase the size of the walkway, passing from 2 to 3. In this case, if the robot finishes in C on the right side (bottom right fig 11) then it is possible to "close" the rectangle by moving down the third non visited column of the walkway, reaching the C' position, and then, the walkway is strictly 2 square large. But, in the last case (bottom left fig 11) it is not possible to finish the walkway with a perfect 2 squares large. We will discuss in the 2 part of the algorithm the consequence.

Part 2 Finishing the coverage

Like shown in figure 9 the finishing coverage is divided in two behaviours for the robot : strait line displacement and boustrophedon displacement. The strait doesn't produce any problem of coverage. First the internal part of the walkway is covered and after reaching the A entry point the external of the walkway is covered until de C point.
After overtaking the C point a boustrophedon displacement is used to finish the covering. Here depending on the parity of the quares of the walkway a non covered square can appear in each time that the walkway turns (see fig 7). An other non covered square can appear, for the same reasons, when the walkway it 3 squares large. So finally in the worse case 3 non covered squares can be obtain. The algorithm is not optimal and no solution exist (fig 12&13).

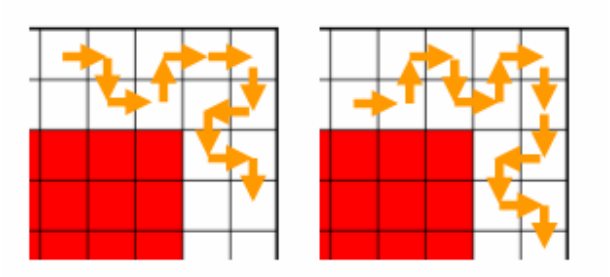

Fig.12. The covering is optimal when the walkway is 2 squares large

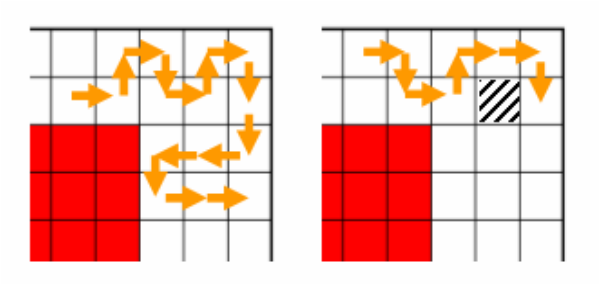

Fig.13. The covering is not optimal when the walkway is 3 squares large it depend on the size.

In this case the robot knows that a square has not been covered because it haves the map of the environment. In this case we can accept that the robot add for instance one backward motion on each of then to finish the coverage.

5. Discussion

In this work we suppose that the robot will exactly move from one square to another. We know that for real robot following a strait line never perfect due to errors introduced by slipping on the floor for instance. For a real application, some trajectory corrections should be added in localisation.
On a second hand, the hypotheses are in figure 3 that the robot is equipped with a line of sensors with 7 elementary obstacle detectors. We did not check it but we believe that only 3 obstacle sensors would enough like in figure.

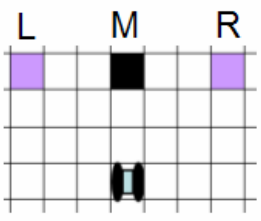

Fig.14. Three sensors should be enough if the obstacles are large

5. Java code for building the map

Here a simplify version of the code used to make the simulation presented in part 3. The full code can be found in [7].

```
void goRobot(JTextArea [][] e,int [][] t,int i,int j){
   int a,fin,odd;
  // Search left
   while ( ! leftObstacle(world,i,j)) {j=j-1; world[i][j]=2;
Red(e,i,j);};
  //draw virtual obstacle in blue
   a=0;
   while(a<20){
    if (world[a][j-3]!=1){world[a][j-3]=3;Blue(e,a,j-3);};
    a++;
   }
  // Search rigth
   i=i-1; world[i][j]=2;Red(e,i,j);a=0; // deplace en haut
   while ( ! murDroite(world,i,j)) {
    if ((world[i+1][j]==0)&&(a==0)){i=i+1;j=j-1;a=1;};
    j=j+1; world[i][j]=2; Red(e,i,j);};
  //blue line j+3
  ...
  // something one left or rigth
   fin=0;odd=0;
   i=i-1;world[i][j]=2;Red(e,i,j);a=0;
   while (fin==0){
    while(! leftObstacle(world,i,j)){
     if (world[i][j-1]==2){i=i-1;j=j+1;};
     j=j-1; world[i][j]=2; Red(e,i,j);
     if (isOdd(world,i,j)){odd=1;};
     if (sensorLeftRigth(world,i,j)){fin=1;} //blue line i-3
    };
    if (fin==0){
     i=i-1;world[i][j]=2;Red(e,i,j);a=0;
     while ( rightObstacle(world,i,j,odd)) {
       if          ((world[i+1][j]==0)&&(a==0)){i=i+1;j=j-
1;a=1;};
      j=j+1; world[i][j]=2; Red(e,i,j); // blue line i-3
     };
    }
   }
  }
```

5. Conclusion

In this paper we have presented a quasi optimal algorithm to cover an unknown environment. We show that the existence of an optimal path depends on the distance between obstacles and the entry/exit doors positions, we show that in the worst case, the all surface is covered we a maximum of 3 elementary square non visited.
The map construction if done with a robot “tank like” with a line of distance sensors of 7 elements, looking 3 steps in front. This kind of robot is very standard in commercial platform.

6. References

[1] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer, 1991.
[2] Heung Seok Jeon, Jung Hwan Park, and Ryumduck Oh. *An Efficient Robot Coverage Algorithm Integrated with SLAM for Unknown Environments* ICAI International Conference on Artificial Intelligence, Las Vegas, july 2010
[3] Maxim A. Batalin and Gaurav S. Sukhatme, *The Analysis of an Efficient Algorithm for Robot Coverage and Exploration based on Sensor Network Deployment*, 2005 IEEE International Conference on Robotics and Automation Barcelona, Spain, April 2005
[4] Howie Choset *Coverage of known spaces: the boustrophedon cellular decompotition*, autonomous robots 9, 247-253 2000
[5]. Sylvia C. Wong Bruce A. MacDonald, *A topological coverage algorithm for mobile robots*, IROS International Conference on Intelligent Robots and Systems 2003, page 1685-1690
[6] E. Prassler, J. Scholz, M. Schuster, and D. Schwammkrug. *Tracking a large number of moving objects in a crowded environment*. In IEEE Workshop on Perception for Mobile Agents, 1998
[7] http://dominique.duhaut.free.fr/sourcefull.java